SQL

Prof. Márcio Bueno

{bd2tarde,bd2noite}@marciobueno.com

Material dos professores Ana Carolina Salgado, Fernando Foncesa e Valéria Times (CIn/UFPE)

SQL

- SQL - Structured Query Language

  Linguagem de Consulta Estruturada

- Fundamentada no modelo relacional, inclui comandos para:
  - definição de dados
  - consulta
  - atualização

SQL

Origem:

- Primeira versão: SEQUEL, definida por Chamberlain em 1974 na IBM
- Em 1975 foi implementado o primeiro protótipo
- Revisada e ampliada entre 1976 e 1977 e teve seu nome alterado para SQL por razões jurídicas

SQL

Origem:

- Em 1982, o American National Standard Institute tornou SQL padrão oficial de linguagem em ambiente relacional
- Utilizada tanto de forma interativa como incluída em linguagens hospedeiras

SQL

Enfoques de SQL:

- Linguagem interativa de consulta (ad-hoc): usuários podem definir consultas independente de programas.
- Linguagem de programação para acesso a banco de dados: comandos SQL embutidos em programas de aplicação.
- Linguagem de administração de dados: o DBA pode utilizar SQL para realizar suas tarefas.

SQL

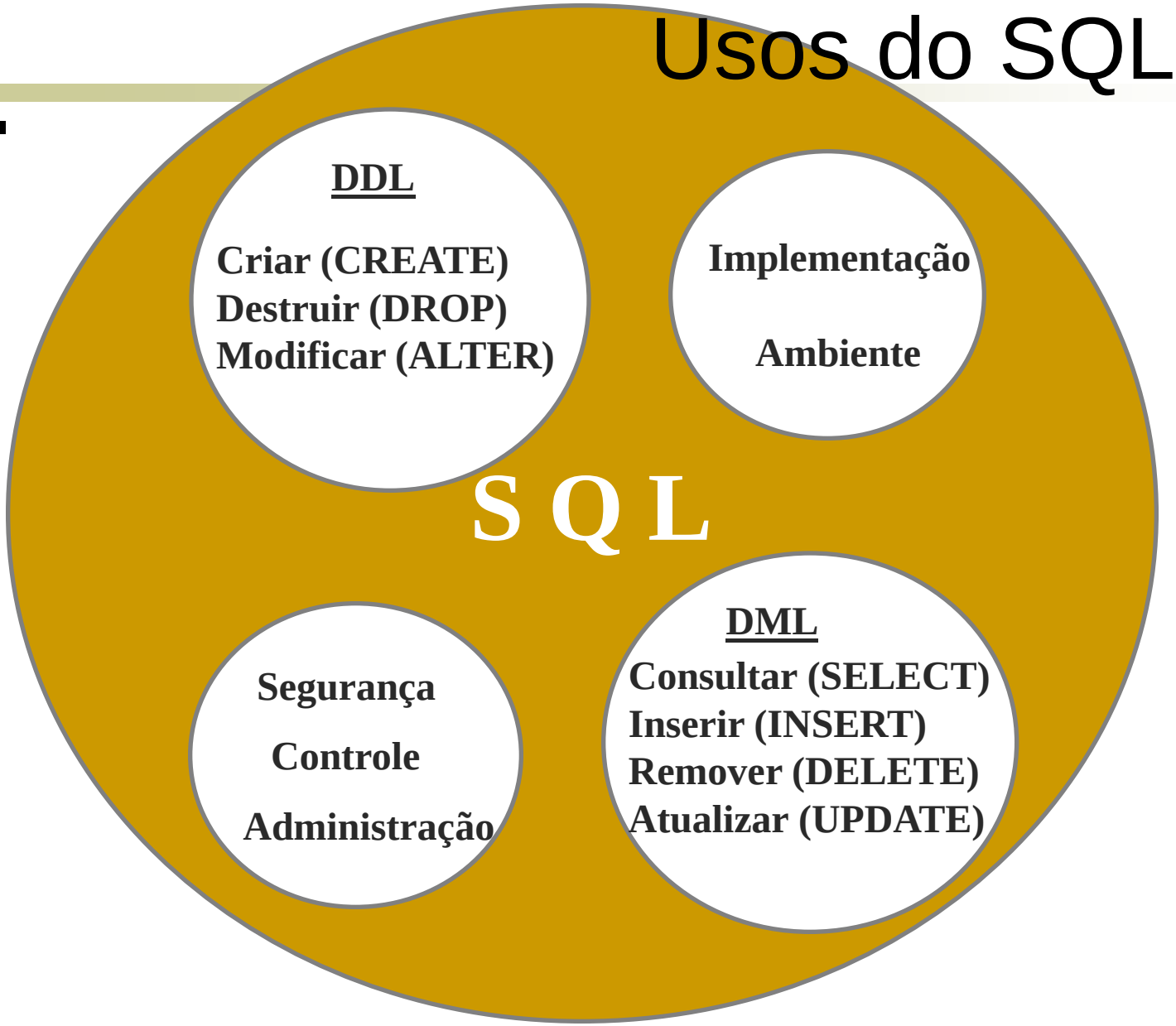
Enfoques de SQL (cont):

- Linguagem cliente/servidor: os programas clientes usam comandos SQL para se comunicarem e compartilharem dados com o servidor.
- Linguagem para banco de dados distribuídos: auxilia na distribuição de dados por vários nós e na comunicação de dados com outros sistemas.

SQL

- Caminho de acesso a outros bancos de dados em diferentes máquinas: auxilia na conversão entre diferentes produtos em diferentes máquinas.

Usos do SQL


DDL
Criar (CREATE)
Destruir (DROP)
Modificar (ALTER)
Implementação
Ambiente
S Q L
Segurança
Controle
Administração
DML
Consultar (SELECT)
Inserir (INSERT)
Remover (DELETE)
Atualizar (UPDATE)

SQL

Vantagens:

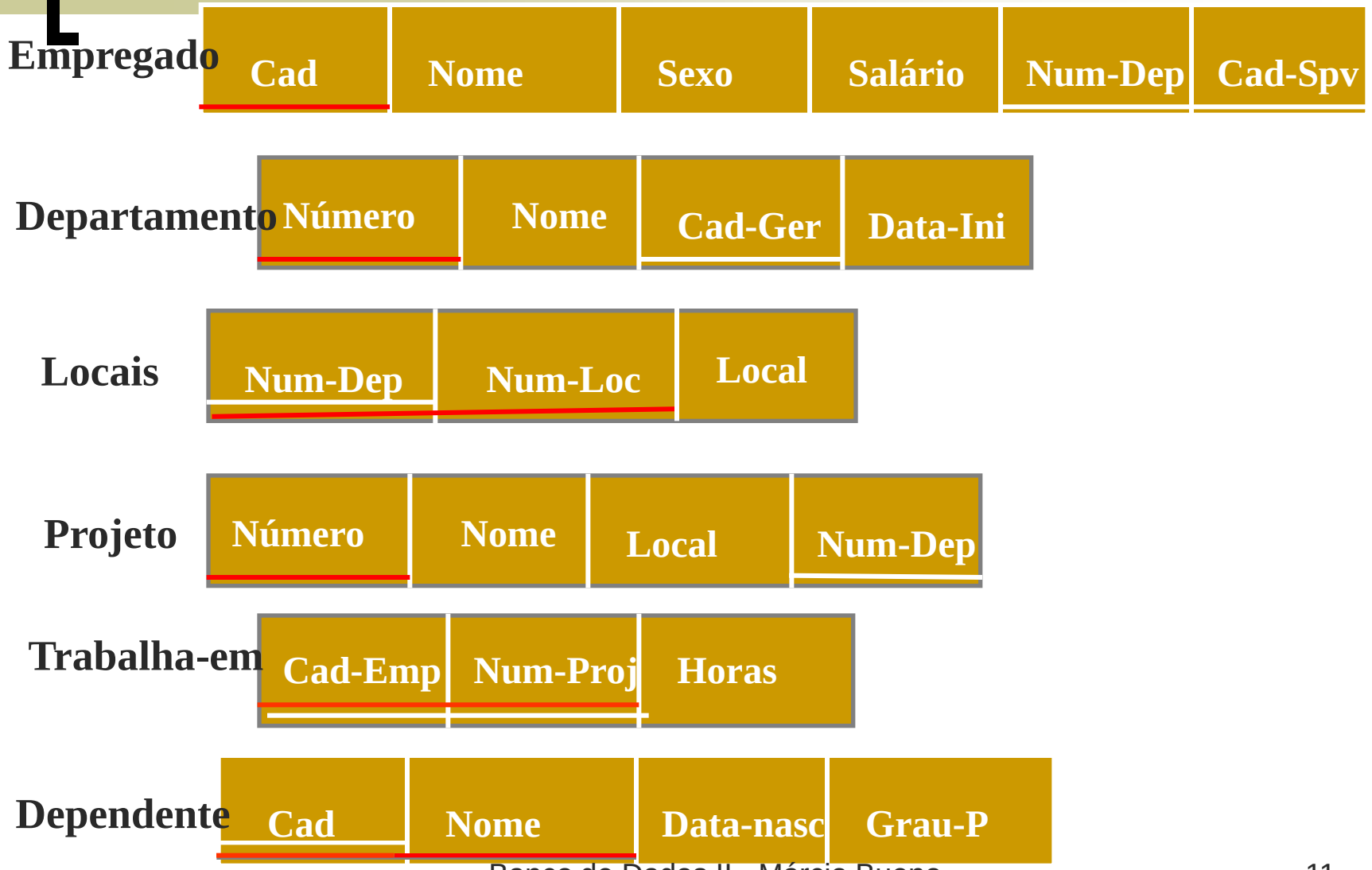
- Independência de fabricante
- Portabilidade entre sistemas
- Redução de custos com treinamento
- Comandos em inglês
- Consulta interativa
- Múltiplas visões de dados
- Manipulação dinâmica dos dados

SQL

Desvantagens:

- A padronização inibe a criatividade
- Está longe de ser uma linguagem relacional ideal
- Algumas críticas:
  - falta de ortogonalidade nas expressões
  - discordância com as linguagens hospedeiras
  - não dá suporte a alguns aspectos do modelo relacional.

Esquema Relacional dos Exemplos



Comandos SQL (padrão ANSI):

- Criação e destruição de tabelas
- Extração de dados de uma tabela
- Inserção, modificação e remoção de dados
- Definição de visões
- Definição de privilégios de acesso

Criação de Tabelas

CREATE TABLE <tabela>

(<descrição dos atributos>

<descrição das chaves>)

- Descrição dos atributos: lista dos atributos com respectivos tipos de dados: smallint, char, money, varchar, integer, decimal, float, real, date, time, timestamp, logical.

SQL

- A chave primária deve ser declarada como not null unique

  ou PRIMARY KEY (<atributos>)
- Descrição das chaves: lista das chaves estrangeiras na forma:

  FOREIGN KEY (<atributo>)
  REFERENCES <tabela> (<atributo>)

Criação de tabelas

Exemplo:

Empregado	Cad	Nome	Sexo	Salário	Num-Dep	Cad-Spv

```
CREATE  TABLE Empregado
  (Cad smallint not null,
    Nome char (20),
    Sexo char,
    Salário decimal (10,2),
    Num-Dep integer,
    Cad-Spv smallint,
   PRIMARY KEY (Cad),
    FOREIGN KEY (Num-Dep) REFERENCES Departamento
   (Número),
    FOREIGN KEY (Cad-Spv) REFERENCES Empregado (Cad))
```

Criação de tabelas

Exemplo:

Trabalha-em


```
CREATE  TABLE Trabalha-em
    (Cad-emp smallint not null ,
     Num-Proj integer not null ,
     Horas decimal (3,1)
     PRIMARY KEY (Cad_emp, Num_proj)
     FOREIGN KEY (Cad-Emp) REFERENCES Empregado
     (Cad),
     FOREIGN KEY (Num-Proj) REFERENCES Projeto
      (Número))
```

Destruição de tabelas

DROP TABLE <tabela>

Elimina a tabela que foi previamente criada.

Exemplo:

DROP TABLE Empregado

Extração de Dados de uma Tabela

Comando SELECT

- Selecionando atributos (Projeção):

  SELECT <lista de atributos> FROM <tabela>

  Exemplo: Listar nome e salário de todos os empregados

  SELECT Nome, Salário FROM Empregado

Extração de Dados de uma Tabela

- Selecionando todos os atributos

  SELECT * FROM <tabela>

  Exemplo: Listar o conteúdo de Empregado

  SELECT * FROM Empregado

Extração de Dados de uma Tabela

- Selecionando tuplas da tabela

  SELECT <lista de atributos> FROM <tabela> WHERE <condição>

  onde condição:

  <nome atributo> <operador> <valor>

Operadores Relacionais: =, <>, <, <=, >, >=

Operadores Lógicos: **AND, OR e NOT**

Extração de Dados de uma Tabela

- Selecionando tuplas da tabela

  Exemplos:

  - Listar nome e sexo dos empregados do departamento 15.

```
SELECT Nome, Sexo FROM Empregado
    WHERE Num-Dep = 15
```

  - Listar nome e sexo dos empregados do departamento 15 com salário > R$ 1.000,00

```
SELECT Nome, Sexo FROM Empregado
    WHERE Num-Dep = 15 AND Salário > 1000
```

Operadores

Operadores BETWEEN e NOT BETWEEN:

substituem o uso dos operadores <= e >=

... WHERE <nome atributo> BETWEEN <valor1> and <valor2>

Exemplo:

- Listar os empregados com salário entre R$ 1.000,00 e R$ 2.000,00

```
SELECT * FROM Empregado
   WHERE Salário BETWEEN 1000 and 2000
```

Operadores

Operadores LIKE e NOT LIKE: só se aplicam sobre atributos do tipo char. Operam como = e <>, utilizando os símbolos: % (substitui uma palavra) e _ (substitui um caracter).

...WHERE <nome atributo> LIKE <valor1>

Exemplo:

- Listar os empregados que tem como primeiro nome 'José'

```
SELECT Nome FROM Empregado
    WHERE Nome LIKE 'José%'
```

Operadores

Operadores IN e NOT IN: procuram dados que estão ou não contidos em um dado conjunto de valores.

... WHERE <nome atributo> IN <valores>

Exemplo:

-Listar o nome e data de nascimento dos dependentes com grau de parentesco ‘M’ ou ‘P’

SELECT Nome, Data-Nasc FROM Dependentes WHERE Grau-P IN (‘M’, ‘P’)

Operadores

Operadores IS NULL e IS NOT NULL:

identificam valores nulos dos atributos.

... WHERE <nome atributo> IS NULL

Exemplo:

- Listar os projetos que não tenham local definido

```
SELECT * FROM Projeto
    WHERE Local IS NULL
```

Ordenação

- Ordenando os dados selecionados (ORDER BY)

  SELECT <lista atributos> FROM <tabela>
  [WHERE <condição>] ORDER BY
  <Nome atributo> {ASC | DESC}

Ordenação

Exemplo:

- Listar todos os empregados ordenados ascendentemente por nome.

```
SELECT * FROM Empregado ORDER BY Nome
```

- Listar todos os empregados ordenados descendentemente por salário

```
SELECT * FROM Empregado ORDER BY Salário DESC
```

Cálculos

- Realizando cálculo com informação selecionada

  Pode-se criar um campo que não pertença à tabela a partir de cálculos sobre atributos da tabela

  Exemplo:

  - Mostrar o novo salário dos empregados calculado com base no reajuste de 60% para os que ganham abaixo de R$ 1.000,00

Cálculos

```
SELECT Nome, Salário * 1.60 AS
    [Novo Salário]
FROM Empregado WHERE Salário < 1000
```

Funções Agregadas

- Utilizando funções sobre conjuntos

  Comandos: MAX, MIN, SUM, AVG, COUNT

  Exemplos:

  - Mostrar o valor do maior salário dos empregados

    SELECT MAX (Salário) FROM Empregado

Funções Agregadas

- Mostrar qual o salário médio dos empregados

  SELECT AVG (Salário) FROM Empregado

- Quantos empregados ganham mais de R$1.000,00?

  SELECT COUNT (*) FROM Empregado WHERE Salário > 1000

Tuplas sem Duplicatas

- Cláusula DISTINCT

  Elimina tuplas duplicadas do resultado de uma consulta.

  Exemplo:

  - Quais os diferentes salários dos empregados?

    SELECT DISTINCT Salário FROM Empregado

Agrupamento de Tuplas

- Agrupando informações selecionadas (GROUP BY)

  Organiza a seleção de dados em grupos

  Exemplo:

  - Listar a média de salário dos empregados categorizadas por sexo

    SELECT AVG(Salário), Sexo FROM Empregado GROUP BY Sexo

Agrupamento de Tuplas

- Agrupando Informações de forma condicional (HAVING)

  Seleciona entre as tuplas resultantes, as que satisfazem um dada condição.

Agrupamento de Tuplas

Exemplo:

- Listar o número total de empregados que recebem salários superior a R$1.000,00, agrupados por departamento mas só daqueles com mais de 5 empregados.

```
SELECT Num-Dep, COUNT (*)
FROM Empregado
WHERE Salário > 1000
GROUP BY Num-Dep
HAVING COUNT(*) > 5
```

Junção de Tabelas

- Recuperando dados de várias tabelas (JOIN)

  Citar as tabelas envolvidas na cláusula FROM

  Qualificadores de nomes: para referenciar o nome do empregado, Empregado.Nome Exemplos:

  - Listar o nome do empregado e do departamento onde está alocado

Junção de Tabelas

```
SELECT
    Empregado.Nome,Departamento.Nome
FROM Empregado JOIN Departamento
ON Empregado.Num-Dep=
            Departamento.Numero;
```

- Listar os nomes dos departamentos que têm projetos

```
SELECT  Departamento.Nome
FROM  Departamento JOIN Projeto
ON Projeto.Num-Dep =
 Departamento.Numero;
```

Junção de Tabelas

- Observações:
  - Podemos utilizar as cláusulas (NOT) LIKE, (NOT) IN, IS (NOT) NULL misturadas aos operadores AND, OR e NOT nas equações de junção.
  - Podemos agrupar e/ou ordenar o resultado de uma junção.
  - Agrupamento/Ordenação pode ser feito através de mais de um atributo.

Junção de Tabelas

- Exemplo: Listar os departamentos que têm projetos com número superior a 99 e localizados em RJ ou SP ordenados por nome de departamento

```
SELECT Departamento.Nome
FROM  Departamento JOIN Projeto
    ON Projeto.Num-Dep = Departamento.Numero
WHERE  Projeto.Local IN ('RJ', 'SP')
    AND Projeto.Número > 99
ORDER BY  Departamento.Nome ;
```

Junção de Tabelas

- Outro Exemplo: Classificando uma Junção
  - Para cada departamento, liste o número, o nome e o salário de seus empregados

```
SELECT D.Nome, E.Cad, E.Nome,
  E.Salario
FROM  Departamento D JOIN Empregado E
  ON D.Numero = E.Num-Dep
ORDER BY  D.Nome, E.Salario DESC;
```

Junção de Tabelas

- Outro Exemplo: Agrupando através de mais de um atributo em uma Junção
  - Encontre o total de projetos de cada funcionário

```
SELECT E.Num-Dep, E.Cad, COUNT(*) AS Total
  FROM  Trabalha-em T JOIN Empregado E
  ON E.Cad = T.Cad-Emp
  GROUP BY  E.Num-Dep, E.Cad
  ORDER BY  E.Num-Dep, E.Cad;
```

Junção de Tabelas

- Juntando mais de duas tabelas

  Exemplo:

  - Listar o nome dos empregados, com seu respectivo departamento que trabalhem mais de 20 horas em algum projeto

Junção de Tabelas

```
SELECT  Empregado.Nome,
    Departamento.Nome
FROM  Empregado JOIN Departamento
   ON Empregado.Num-Dep  =
    Departamento.Número
    JOIN Trabalha-em
     ON Trabalha-em.Cad-Emp =
    Empregado.Cad
WHERE  Trabalha-em.Horas > 20
```

Consultas Encadeadas

- Utilizando consultas encadeadas
  - O resultado de uma consulta é utilizado por outra consulta, de forma encadeada e no mesmo comando SQL.
  - O resultado do comando SELECT mais interno (*subselect*) é usado por outro SELECT mais externo para obter o resultado final.
  - O SELECT mais interno (*subconsulta* ou *consulta aninhada*) pode ser usado principalmente nas cláusulas WHERE e HAVING do comando mais externo.

Consultas Encadeadas

- Utilizando consultas encadeadas(Cont.)
  - Existem 3 tipos de subconsultas:
    - ESCALAR: Retornam um único valor.
    - ÚNICA COLUNA: Retornam várias linhas, mas apenas uma única coluna é obtida.
    - TABELA: Retornam uma ou mais colunas e múltiplas linhas.

Consultas Encadeadas

- Exemplo: Usando uma subconsulta com operador de igualdade
  - Listar os empregados que trabalham no departamento de Informática.

```
SELECT  Cad, Nome, Salario
FROM  Empregado
WHERE Num-Dep  =
                    (SELECT  Numero
                     FROM  Departamento
                     WHERE Nome = 'Informática' );
```

Consultas Encadeadas

- Observações
  - A subconsulta anterior é do tipo ESCALAR: Retorna o código do departamento de Informática.
  - Operadores Relacionais: =, <>, <, <=, >, >=
  - A subconsulta deve sempre estar entre ().

Consultas Encadeadas

- Exemplo com função agregada
  - Listar os empregados cujos salários são maiores do que o salário médio, mostrando o quanto são maiores.

```
SELECT  Cad, Nome, Sexo,
     Salario - ( SELECT AVG (Salario)
                 FROM  Empregado)  AS  DifSal
FROM  Empregado
WHERE Salario  >  ( SELECT AVG ( Salario)
                    FROM  Empregado );
```

Consultas Encadeadas

- Observações
  - A subconsulta anterior é do tipo ESCALAR: Retorna o salário médio dos empregados.
  - A condição abaixo é incorreta:

    WHERE Salario > AVG (Salario)

Consultas Encadeadas: IN

- Listar os departamentos que tenham qualquer projeto em RJ

```
SELECT  Departamento.Nome
  FROM Departamento
  WHERE  Departamento.Número IN
          (SELECT Projeto.Num-Dep
             FROM Projeto
       WHERE Projeto.Local = ‘RJ’) ;
```

Consultas Encadeadas: IN

- Listar os dependentes dos funcionários que trabalham no departamento de Informática

```
SELECT  Nome, Data-nasc, Grau-P
FROM Dependente
WHERE  Cad IN  ( SELECT Cad
                 FROM Empregado
                 WHERE Num-Dep =
                (SELECT Numero
                 FROM  Departamento
                 WHERE Nome = 'Informática' ) );
```

Consultas Encadeadas: ANY/SOME

- São usadas com subconsultas que produzem uma única coluna de números.
- Listar os empregados cujos salários são maiores do que o salário de pelo menos um funcionário do departamento 20

```
SELECT  Cad, Nome, Sexo, Salario
FROM Empregado
WHERE Salario > SOME (SELECT Salario
                      FROM Empregado
                      WHERE Num-Dep =  20) ;
```

Consultas Encadeadas: ALL

- É utilizado com subconsultas que produzem uma única coluna de números.
- Listar os empregados cujos salários são maiores do que salário de cada funcionário do departamento 15

```
SELECT  Cad, Nome, Sexo, Salario
FROM Empregado
WHERE Salario > ALL ( SELECT Salario
                      FROM Empregado
                      WHERE Num-Dep =  15) ;
```

Consultas Encadeadas

- Regras genéricas de subconsultas
  - A cláusula ORDER BY não pode ser usada em uma subconsulta.
  - A lista de atributos especificados no SELECT de uma subconsulta deve conter um único elemento (exceto para EXISTS).
  - Nomes de atributos especificados na subconsulta estão associados às tabelas listadas na cláusula FROM da mesma.
    - É possível referir-se a uma tabela da cláusula FROM da consulta mais externa utilizando qualificadores de atributos.

Consultas Encadeadas

- Regras genéricas de subconsultas (Cont.)
  - Quando a subconsulta é um dos operandos envolvidos em uma comparação, ela deve aparecer no lado direito da comparação.
    - Por exemplo, seria incorreto especificar:

```
SELECT  Cad, Nome, Sexo, Salario
FROM  Empregado
WHERE
      ( SELECT AVG ( Salario)
        FROM  Empregado )  <  Salario;
```

Consultas Encadeadas

- EXISTS e NOT EXISTS
  - Foram projetadas para uso apenas com subconsultas.
  - EXISTS
    - Retorna TRUE ⇔ existe pelo menos uma linha produzida pela subconsulta.
    - Retorna FALSE ⇔ a subconsulta produz uma tabela resultante vazia.

Consultas Encadeadas

- Exemplo: Utilizando EXISTS
  - Liste todos os empregados que trabalham no departamento de Informática.

```
SELECT  Cad, Nome, Sexo, Salario
   FROM  Empregado E
   WHERE EXISTS
       ( SELECT   *
          FROM  Departamento D
         WHERE  E.Num-Dep  =  D.Numero AND
                        D.Nome =  'Informática') ;
```

Operações de Conjunto

- UNION
  - Linhas duplicadas são removidas da tabela resultante.
  - Exemplo: Construa uma lista de todos os locais onde existe um departamento ou um projeto.

```
( SELECT  Local  FROM  Projeto
  WHERE Local IS NOT NULL )
UNION
   ( SELECT Local  FROM  Locais ) ;
```

Operações de Conjunto

- INTERSECT
  - Exemplo: Construa uma lista de todos os locais onde existe ambos um departamento e um projeto.

```
( SELECT Local
  FROM Projeto )
INTERSECT
   ( SELECT Local
     FROM Locais ) ;
```

Operações de Conjunto

- EXCEPT / MINUS
  - Exemplo: Construa uma lista de todos os locais onde existe um departamento mas nenhum projeto.

```
( SELECT Local
  FROM  Locais )
EXCEPT
   ( SELECT  Local
     FROM  Projeto );
```

Inserção de Tuplas

- Adicionando tupla à tabela

INSERT INTO <tabela> (<lista de atributos>)
VALUES (<valores>);

INSERT INTO Empregado (Cad, Nome, Sexo, Salário, Num_Dep, Cad_Supv)
VALUES (015, ‘José da Silva’, ‘M’, 1000, 1, 020)

Inserção de Tuplas

- Adicionando tuplas usando SELECT

INSERT INTO <tabela> (<lista de atributos>)
SELECT <lista de atributos>
FROM <tabela>
WHERE <condição>

Inserção de Tuplas

```
INSERT  INTO depto-info (nome-depto, num-
    emp, total-sal)
SELECT D.nome, COUNT(*), SUM
    (E.salario)
FROM  Departamento D JOIN Empregado E
     ON D.numero  = E.Num-Dep
GROUP BY D.nome
HAVING COUNT (*)  > 50 ;
```

Alteração de Tuplas

- Atualizando tuplas

```
UPDATE <nome tabela>
SET <nome atributo> = valor
WHERE <condição> ;

UPDATE Empregado
SET Salário = 1500
WHERE Cad = 015;
```

Removendo Tuplas

- Apagando tuplas da tabela

```
DELETE  FROM <tabela>
WHERE <condição> ;
```

```
DELETE  FROM Empregado
WHERE  Salário > 5000 ;
```

Visões

- Utilizando Visões (VIEWS)

  São tabelas virtuais que não ocupam espaço físico

  - Criação, Utilização, Inserção, Modificação, etc.

    CREATE VIEW <nome da view> <lista de atributos> AS SELECT...

Visões

- Criar uma visão dos empregados do departamento 10 que tenham mais de 20 horas de trabalho em projetos

```
CREATE VIEW Dep-10 AS
SELECT  Empregado.Nome,
    Trabalha-em.Num-Proj
FROM  Empregado JOIN Trabalha-em ON
    Trabalha-em.Cad-Emp = Empregado.Cad
WHERE Trabalha-em.Horas > 20
   AND Empregado.Num-Dep = 10;
```

Sumário

- Forma Geral de um Comando SQL:

SELECT [DISTINCT / ALL] { * | expressão
[AS novoNome] | listaAtributos }

FROM nomeTab1[alias1] [JOIN]
nomeTab2[alias2] [ON condição] ...

[WHERE condição]

[GROUP BY listaColunas] [HAVING
condição]

[ORDER BY listaColunas]

Sumário

- Cada comando SQL produz uma tabela resultante consistindo de 1 ou mais colunas e de 0 ou mais linhas.
- Cláusula SELECT
  - Identifica colunas e/ou dados derivados que aparecerão na tabela resultante.
  - Todos os atributos devem ter suas tabelas/visões correspondentes, listadas na cláusula FROM.
  - SELECT e FROM são as únicas cláusulas obrigatórias.

Sumário

- Cláusula WHERE: Seleciona linhas para serem incluídas na tabela resultante.
- Cláusula ORDER BY
  - Permite ordenar a tabela resultante com base nos valores de 1 ou mais atributos.
  - Pode ser feita de forma crescente ou decrescente.
  - Se especificada, deve ser a última cláusula do comando SQL.

Sumário

- Funções COUNT, SUM, AVG, MIN e MAX
  - Recebem uma coluna inteira como argumento e retornam um único valor.
  - COUNT, MIN e MAX aplicam-se a atributos numéricos/ não-numéricos.
  - É ilegal misturar estas funções com nomes de atributos do SELECT, exceto se a cláusula GROUP BY é usada.
  - Valores nulos são eliminados por cada função (exceto no caso de COUNT(*)).
  - Para eliminar duplicadas, use DISTINCT.

Sumário

- Cláusula GROUP BY
  - Permite que informação de resumo seja incluída na tabela.
  - HAVING atua como uma cláusula WHERE para grupos, restringindo-os.
    - Porém, HAVING pode ter funções agregadas.